

Paul Johnson
POLS 707 Research Methods II

Top 13 Things worth knowing about R.

1. S books are R books.

S is a statistical language developed at Bell Labs (R.A. Becker, J.M. Chambers, A.R. Wilks, *The New S Language*. Pacific Grove, CA: Wadsworth & Brooks/Cole, 1988) and a computer program that speaks S, called “S” (or, more recently, “S+”) is for sale from Mathsoft, Inc.

R is a statistical language very much like S, better in some ways, and it is available for free and in open source code. R is created/developed by a community of statisticians and programmers who truly desire to make good tools available for research and escape the binds imposed by commercialization of software.

Books about S generally will also be applicable to R. Here are some of the really useful ones of which I have copies.

W.N. Venables and B.D. Ripley, *Modern Applied Statistics with S*, 4th edition. Brian Ripley is an amazing guy. He must work 20 hours per day. He is one of the primary developers behind the R movement and he reads the r-help email list. If you ever touch on some problem deep down in the guts of R, he may be the only one who will really know the answer.

Peter Dalgaard, “Introductory Statistics with R”. Peter Daalgard is very active in r-help and also he is a very pleasant person and a good writer. This book is very well done. Not so advanced as V&R, but much more understandable to your average user. This shows how to complete many “garden variety” tasks.

Frank E. Harrell, Jr. *Regression Modeling Strategies: With Applications to Linear Models, Logistic Regression, and Survival Analysis*. This book has great practical information about how to do projects. He has complete code and working examples. Especially important if you want to use Harrell’s advanced R packages, Hmisc and Design. Frank Harrell is (like me) a “refugee” from SAS and S+. He was the one who wrote the first “Proc Logistic” that was available in SAS in the early 1980s. In the 1990s he was working in S and S+.

The S-Plus stat manuals are available for free, in pdf format!

S-Plus site: <http://www.insightful.com/support/documentation.asp?DID=3>

You especially want the Guide to Statistics, Parts I and II. These have simpler explanations of many models than you will find in other places. The syntax for usage is not exactly the same as R, but they are worth downloading and saving for reference.

2. R has built-in help.

- (a) Procedure specific help. If you want to know more about the “hist” procedure, type
> help(hist)

That “help” method gets used so often that they created a short-cut for it:

```
> ?hist
```

and after you look that over, you notice in the bottom of the help page it has examples, and if you want to see the examples, do

```
> example(hist)
```

If the examples all whir by too quickly, use this command to cause R to ask you to hit the return key between pictures:

```
> par(ask=T)
```

and then type your `example(hist)` command again.

- (b) Get help inside your web browser. Type

```
> help.start()
```

and watch what happens. Your web browser should show you a top level view, which includes links to the “Introduction to R” book as well as the manual on data import and export.

You will see also there is a FAQ for R in the online docs. The R FAQ is not a list of precise details about how to do certain things, but rather a higher level explanation of the R language and the various platforms on which it is used.

- (c) Type

```
> help.search(“hist”)
```

and watch what happens. This returns a giant list of methods, most of which you don’t really want. But some you do. Alternative versions of histogram methods appear. For example, Brian Ripley does not like the default “hist” method, so his package MASS includes “truehist”. The MASS package’s truehist is more readily configurable.

3. Every R command needs parentheses!

Even when you want out of R, you can’t just type quit.

You type

```
>quit()
```

or, for short,

```
> q()
```

If you type

```
> q
```

R thinks you want to print out the contents of something named “q”, but it can’t find such a thing, and it tells you so.

4. “Equal to” means a whole different thing than you expect.

If you create a new variable, do not use the equal sign (=). Instead, you must use the symbol < –to “assign” a value. So, for example, you can set a constant named “b”:

```
> b < -3.3
```

or define a vector named “b”:

```
> b < -c(3,2,1,4,1,5,5,1,1,2,)
```

or, if you run some command, like `read.table()`, you can save its result as a data frame named “b”

```
> b <- read.table("myData.txt",header=T)
```

Note the equal sign is used inside the parentheses. The equal sign is OK there, because it is not making a permanent assignment to some variable.

If you study other math-oriented computer packages, such as *Mathematica*, you will find the same kind of distinction between equals and assignment. Its not an R thing in particular.

5. Terminology:

I only bring these things up because you are likely to get confused if you go into some manual and find that they are using a lot of terms that are unfamiliar to a social scientist.

(a) object: a “thing” that

i. has variables (one or more): that means it has a “name” and a “value” for each variable. The value can be a missing value.

ii. can do “stuff” that you ask it to.

(b) method: the instructions that are sent to objects.

I frequently betray my SAS background by calling these things “procedures”.

In R, the style is different than I was used to from computer languages like Java and Objective-C. (It seems to me that, in R, one thinks of a message sent to an object in a way that is almost exactly the opposite of Java.) If you ask an object to carry out a method, in R you say:

```
> method(object)
```

whereas in java you would say:

```
object.method()
```

(c) In R, you can do things however you like, but there is a certain recommended style. Most importantly, always remember that the results of methods are always objects.

You can run a regression model and watch the output on the screen with this sort of command:

```
lm(y~x)
```

but most people don’t do that. Rather, they save the “regression object” and then use it:

```
myRegObj <- lm(y~x)
summary(myRegObj)
```

summary is the method, which the object myRegObj carries out. The R style looks

like a function call, but it is not, really.

You see my Objective-C background pop out here because I like to name things verbosely, starting with a small letter, then using capitals to start new elements of the name.

6. In R, you can type in lots of stuff. But you should not.

Write your commands into a text editor and then run them in R. This way, you have a perfect record of what you have done and you can always reproduce it perfectly.

The editor Emacs (and Xemacs) can be customized to work together with R, but I don't think there is a big advantage for most users from doing that.

7. R has what our students need.

- (a) The standard R distribution includes all statistical methods (and many more) that are used in an intermediate course like POLS 707. For linear models, use `lm()`. For scatterplots, use `plot()`. For histograms, start with `hist()` (or `truehist()`). For logit models, use `glm()`. For factor analysis, use `aov()`.
- (b) R includes a general purpose programming language, featuring vectors, matrices, and many other excellent things. There is no limit to the complexity of your projects (except the limits imposed by your patience, intelligence, and access to hardware).

8. Factors!

R methods are (usually) sensitive to the kinds of variables you ask them to use. They will automatically treat categorical variables differently from continuous variables.

Understand this term:

Factor. S/R terminology for a variable that is not continuous. Factors are “grouping variables” and they can be ordered (some, lots, all) or unordered (male, female).

In the “old days”, the statistical software would not try to separate variables that are meaningfully scaled (age, income, etc) from variables that were not. The user had the duty of creating a coding scheme for categorical variables. Remember “recoding” simply to force categorical variables into a numerical framework, so they could be put into models?

Example 1: `dem = 1` if the respondent is a Democrat, 0 otherwise,

Example 2: `vote = 1` if the respondent voted, 0 otherwise.

In R, if you input some variable with values like “D” and “R” and then you declare it as a factor, then R will “automagically” recode it for you (turn it into 0 and 1). There are a few different ways it can automagically recode categorical variables. Look for “contrasts” in the documentation. If a model does not make sense, then R will tell you so.

Check this out:

```
> sex <- c("M", "F", "M", "F", "M", "F", "M", "F")
> age <- c(2,3,1,4,2,4,2,1)
> #lm(age~sex) returns an error, but this does not
> sex <- as.factor(sex)
> lm(age~sex)
```

The `lm` method automatically creates a dummy variable “sexM” that is used in the regression model $y = \beta_0 + \beta_1 \text{sexM}$.

Now change sex:

```
> sex <- c("M", "F", "M", "F", "M", "F", "I", "I")
> sex <- as.factor(sex)
> lm(age~sex)
```

That automatically creates 2 dummy variables, “sexM” and “sexI” and it estimates $y = \beta_0 + \beta_1 \text{sexM} + \beta_2 \text{sexI}$

If you already have the numerical data, you tell R to use it as a factor with the “factor” method and then you can set the labels with the “levels” method.

```
> sex <- c(0,1,0,1,0,1,0,1)
> factorSex <- factor(sex,levels=0:1)
> levels(factorSex) <- c("M", "F", "I")
> factorSex
[1] M F M F M F M F
Levels: M F I
```

I do not know if S was the first language to explicitly build in the cautious treatment of categorical variables, but I’m pretty sure it was before SAS in that regard. Now SAS has the “CLASS” option in some procedures. It works like factors in R.

9. Graphics: The agony and the extacy.

(a) Device. This terminology baffled me for a long time. I tend to think of device as a machine. But that’s not what R means by device. Device means an output format. The screen output device `x11()`. If you want to write the putput in a postscript file, use `postscript()`. There are devices for `jpg`, `png`. In MS Windows, there is a device to save windows meta file format.

(b) In Rtips I give some details about saving graphs. I think it is a pain, frankly, but here is the story. If you make a graph on the screen, you can’t always save it into a file so it looks just the same. To be save, it is necessary to fiddle the graph on the screen, and then turn on an output device that writes a file, and then re-run the graph command.

10. There is some stuff in R-base, but there is much much more in R addon packages. If you want to use those packages, you have to explicitly load them with the `library()` command.

R is an “extensible” product, meaning that users can freely write additional capability, package it up, and distribute it. When I first heard of R, there was complete *package anarchy*. Now there is a recommended set of packages that the R makers put together with R when it is distributed, and on the R Internet nexus called “CRAN” one can find many more packages that work, to varying degrees.

Example. “truehist” shows up in the output from `help.search(“hist”)`:

```
truehist(MASS) Plot a Histogram
```

This means that the `package(MASS)` (guess what that’s short for) has another histogram method. In order to use that, you must load the package with this command

```
> library(MASS)
```

And then you can read the help page:

```
> ?truehist
```

In that writeup, you see that you get access to many more settings than you get with the ordinary `hist` method.

Here are packages that I will install and will be available in our stat lab

`car`: a companion for applied regression by John Fox, who has published a several textbooks on regression modeling, the most recent of which is from Sage, *R and S+ Companion to Applied Regression*.

`Rcmdr`: a gui interface for R, also by John Fox. Type `> library(Rcmdr)` to start it up. It is being actively updated. I notice it gets better all the time. Also from John Fox.

`mgcv`: a package for generalized additive regression and Smoothing models

`Hmisc,Design`: packages from Frank Harrell, a biomedical stats professor/researcher who previously had written procedures for SAS and S+.

I will install other packages if you ask nicely and I agree they are needed, otherwise you can install packages in your private user space. R has documentation for how to do that.

11. Model notation in R. The literature on the “generalized regression model” introduced a streamlined notation for regression models that is followed in R and some other languages. It is called “Wilkinson and Rogers” notation and R’s notation is very close to W&R. It is described on p. 75-79 in *An Introduction to R*. If you want to estimate a model:

$$y_i = \alpha + \beta_1 x_{1i} + \beta_2 x_{2i}$$

Then the formula notation for that in R is

$$y \sim x1 + x2$$

Typically, the software will assume you want an estimate for the constant (α) and that you want to estimate separate coefficients (β_1 and β_2) for variables x_1 and x_2 .

Suppose you tell R you want to estimate the formula:

$$y \sim x1 * x2$$

That will estimate a model that includes independent variables $x1$ and $x2$ as well as the “interaction” $x1 * x2$:

$$y_i = \alpha + \beta_1 x1_i + \beta_2 x2_i + \beta_3 x1_i * x2_i$$

If you happened to enter a formula like $x1 * x2 * x3 * x4 * x5$, R would estimate a LOT of coefficients, because you’d get an estimate for each variable, each pair of variables multiplied together, each set of 3 variables multiplied, and so forth.

There is another especially convenient element in R. You can use any of the mathematical functions that R has *inside* your formula. For example, a formula like:

```
> logReg <- lm (y ~ x1+log(x2))
```

will provide estimates for a model:

$$y_i = \beta_0 + \beta_1 * x1 + \beta_2 * \log(x2)$$

I posted an example program “easyLogReg.R” to demonstrate that. You can find it in my R ExampleCode directory
<http://www.ku.edu/~pauljohn/R/ExampleCode>

12. With R, you can do many things that other stat frameworks would not allow or facilitate.
 - (a) In Stata and SPSS, you can only open one data set at a time. That’s a crippling limitation, in my opinion.
 - (b) R is similar to SAS in the sense that one can run a model, save its results into various new datasets, and then do additional work with those new datasets. However, R is orders of magnitude more convenient. Compare the R code to illustrate the Central Limit Theorem with the mean of a Gamma variable
http://www.ku.edu/~pauljohn/R/ExampleCode/template_gamma4.R
against the SAS code for the same purpose:
<http://lark.cc.ku.edu/~pauljohn/SASClass/ExampleCode/clt-gamma2.sas>
 - (c) R has great facilities to handle repetitive tasks. I have run simulations that generate hundreds of output data sets. I have written R code that can systematically open each one, make plots, and calculate results. If I had to do that by hand, I’d be as insane as most Windows users.
13. You can join the r-help email list and you can ask for help in there. Heed my warning. Many people will be helpful to you, but if your question betrays a total lack of effort to read the easily available documentation, then they will not be so eager to help.

I find the r-help list is most helpful for relatively specific questions about how some specific command R can be used or a specific malfunction that you have observed. Nobody is interested in a general email like “I can’t make linear models work in R”, but they are eager to help if you ask a specific question about the syntax in the method `lm()`. They are also eager to point your attention to packages that provide certain functions, but they’ll urge you to search on CRAN before asking. For example, if you search on CRAN and see 4 packages that can handle “multiple imputation for missing data,” it would be suitable to join r-help and ask if anybody will share their experiences with these packages.

If you want to ask in the r-help email list, you should follow this progression.

- (a) Run `help.search(“your topic”)` to see what pops up.
- (b) Consult “Introduction to R,”
- (c) Check the Venables & Ripley book. Check the Daalgaard book.
- (d) Look through my Rtips collection of usage tidbits. That’s what I do, and when I learn something new, I often put it in there. Or at least I did a while ago: Rtips: <http://www.ku.edu/~pauljohn/R/Rtips.html>
- (e) Go to the R email list online archive and search for some key words. Chances are, if you have a question, some other person already had it. You can find about the R mailing lists and their archives on the main R web site: <http://www.r-project.org/>. If you go to this page, which also keeps email list archives, <http://maths.newcastle.edu.au/~rking/R/> at the bottom of the page you will find a SEARCH tool that is rather convenient.